Europäisches Patentamt	European Patent Office	Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

02425471.6

USAN
10/623,146

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 02425471.6
Demande no:

Anmeldetag:
Date of filing: 19.07.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

STMicroelectronics S.r.l.
Via C. Olivetti, 2
20041 Agrate Brianza (Milano)
ITALIE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

A process for managing system stacks in microcontrollers, corresponding device and computer program product

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/ Classification internationale des brevets:

G06F9/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

For the original title, see page 1 of the description

"Procedimento per la gestione di stack di sistema in microcontrollori, relativo dispositivo e prodotto informatico"

* * *

Campo dell'invenzione

La presente invenzione si riferisce alla gestione dello stack di sistema nei microcontrollori ed è stata sviluppata con particolare attenzione alla possibile applicazione alla gestione di uno stack di sistema in un microcontrollore, ad esempio del tipo register file based. Il riferimento a tale possibile applicazione non deve peraltro intendersi come limitativo della portata dell'invenzione, che è affatto generale.

Descrizione della tecnica nota

Una tipica operazione eseguita con grande frequenza nei microcontrollori è la memorizzazione/ripristino dello stato del sistema in particolari locazioni di memoria volatile (generalmente RAM). Tale operazione deriva direttamente dalla necessità di eseguire salti verso apposite routine in istanti arbitrari, all'interno del flusso di programma, in seguito al verificarsi di interrupt esterni o interni.

Lo stato del sistema consiste in un certo numero di registri, diverso per ciascun microcontrollore. Tipicamente lo stato è ricostruibile utilizzando i registri Program Counter (PC) e Condition Code Register (CCR) in aggiunta ad un numero variabile di altri registri.

L'operazione di memorizzazione/ripristino dello stato può quindi impegnare il microcontrollore per un intervallo di tempo dedicato alla funzione di "context switching". Tale intervallo è di solito abbastanza lungo a causa della sequenzialità con cui avviene la scrittura/ripristino dello stato nello stack di

sistema. Ciò determina un decadimento delle prestazioni complessive, anche in applicazioni di controllo non in tempo reale.

Molto spesso, poi, i registri non hanno dimensioni pari a multipli della lunghezza della parola di memoria volatile in cui sono destinati ad essere memorizzati. Tipico esempio di tale situazione è la memorizzazione del registro CCR, che consiste normalmente di pochi bit.

La situazione in una memoria RAM a seguito di una o più esecuzioni di diversi interrupt può quindi essere quella rappresentata - a titolo di esempio - nella figura 1 dei disegni allegati, dove il riferimento U indica locazioni di memoria non utilizzate (unused).

La situazione descritta corrisponde ad uno spreco di memoria dovuto alla ridotta dimensione del registro CCR, o di qualsiasi altro registro di stato, tale da presentare dimensioni ridotte rispetto alla lunghezza della parola della memoria RAM.

Ricorrendo ad una soluzione tradizionale, quale quella illustrata a titolo di esempio nella figura 2, la gestione dello stack di sistema all'interno della memoria RAM, indicato con ST, avviene in modo sequenziale, senza ottimizzazione dello spazio utilizzato dallo stack di sistema in memoria.

L'architettura tradizionale rappresentata nella figura 2 prevede che un'unità di controllo UC generi due segnali use_ss e up_down indirizzati verso l'unità SM fungente da gestore di stack. L'unità SM genera il puntatore allo stack di sistema, correntemente denominato System Stack Pointer o ssp, destinato ad essere inviato verso un primo modulo 10.

Il modulo 10 funge da multiplexer che seleziona tra il segnale ssp, durante il context switching, e il segnale ram_pointer, indicativo dell'indirizzo RAM,

durante il funzionamento normale, così da generare un segnale address_ram utilizzato per la gestione della parte di memoria RAM dedicata agli indirizzi.

Un altro modulo 12 con funzione di multiplexer riceve in ingresso, oltre al segnale data_bus proveniente dal bus dati del microcontrollore, anche due segnali PC e CCR rispettivamente indicativi del Program Counter e del Condition Code Register. Tutto questo per generare in uscita il segnale data_ram utilizzato per la gestione della parte di memoria RAM dedicata ai dati.

I riferimenti 10a e 12a indicano le linee su cui sono presenti i segnali di selezione dei multiplexer che regolano il funzionamento dei moduli 10 e 12, mentre le linee indicate con wen e csn corrispondono alle linee su cui sono inviati verso lo stack ST i corrispondenti segnali indicativi rispettivamente dell'abilitazione alla scrittura ed alla lettura della memoria RAM dove risiede lo stack di sistema.

Sulla linea di uscita dello stack, indicata con 14, è presente il relativo segnale di uscita, indicato con out_stack.

L'architettura rappresentata nella figura 2 corrisponde a soluzioni di tipo noto, il che rende superfluo fornire in questa sede una descrizione di maggior dettaglio.

Tale soluzione è impiegata, ad esempio, nel microcontrollore venduto con il nome commerciale di ST 5 dalla stessa Richiedente. Si tratta per la precisione di un microcontrollore del tipo denominato register file based.

Lo svolgimento dell'operazione di context switching cui si è fatto cenno in precedenza richiede tipicamente un intervallo di tempo pari a circa due periodi di clock per un microcontrollore del tipo ST 5

e circa dieci periodi di clock per un microcontrollore del tipo denominato accumulator based.

Scopi e sintesi della presente invenzione

La presente invenzione si prefigge lo scopo di superare gli inconvenienti intrinseci della soluzione secondo la tecnica nota cui si è fatto riferimento in precedenza riducendo quindi tanto lo spreco di memoria quanto i tempi di esecuzione del context switching.

Secondo la presente invenzione, tale scopo è raggiunto grazie ad un procedimento per la gestione di stack avente le caratteristiche richiamate in modo specifico nelle rivendicazioni che seguono. L'invenzione riguarda anche il corrispondente dispositivo nonché il corrispondente prodotto informatico direttamente caricabile nella memoria interna di un elaboratore numerico e comprendente porzioni di codice software suscettibili di attuare il procedimento secondo l'invenzione quando il prodotto è eseguito su un elaboratore numerico.

Nella forma di attuazione al momento preferita, la soluzione secondo l'invenzione prevede di utilizzare due segnali di abilitazione per due diversi gestori di stack. Il ricorso a questa soluzione fa sì che, nel caso il microcontrollore debba eseguire, ad esempio, un'istruzione di chiamata (call), si abbia la necessità di memorizzare/ripristinare un numero inferiore di informazioni. Tutto questo con la possibilità di non accedere ad uno dei due stack. Per un'istruzione quale un'istruzione di call non è infatti necessario memorizzare il registro CCR.

Il fatto di avere due stack ai quali si accede in modo concorrente migliora le prestazioni in termini di tempo necessario per il context switching, di fatto dimezzandolo. Inoltre, la possibilità di utilizzare diversi tipi di memoria per contenere lo stack di

sistema permette anche di migliorare l'utilizzo delle risorse: stack realizzati con elementi di memoria le cui dimensioni di riga possono essere diverse fra loro riducono lo spreco di locazioni altrimenti inutilizzate.

La soluzione secondo l'invenzione è di validità generale, poiché applicabile, ad esempio, tanto a microcontrollori del tipo accumulator based quanto a microcontrollori del tipo register file based. La soluzione secondo l'invenzione è altresì indipendente dal numero di registri memorizzati durante il context switching e riduce il tempo relativo migliorando le prestazioni complessive del microcontrollore. In più, la soluzione secondo l'invenzione ottimizza l'utilizzazione della memoria per lo stack di sistema.

Breve descrizione dei disegni annessi

L'invenzione sarà ora descritta, a puro titolo di esempio non limitativo, con riferimento ai disegni annessi, nei quali:

- le Figure 1 e 2, inerenti alla tecnica nota, sono già state descritte in precedenza, e

- la Figura 3 illustra, con un formalismo mirante a permettere il diretto confronto con la soluzione nota illustrata nella figura 2, un'architettura per la gestione di stack di sistema operante secondo la presente invenzione.

L'esempio di attuazione dell'invenzione rappresentato nella figura 3 si basa essenzialmente sulla soluzione di "sdoppiare" l'architettura rappresentata nella figura 2.

Ferma restando la presenza di un'unità di controllo UC destinata a generare il segnale up_down che discrimina fra un'operazione di "push" o di "pop" (questa terminologia è di significato evidente per i tecnici del settore), è prevista la presenza di due

gestori di stack distinti, indicati rispettivamente con SM1 e SM2. L'unità di controllo UC invia verso i gestore SM1 e SM2 due rispettivi segnali use_ss1 e use_ss2, ciascuno dei quali è destinato ad indicare - rispettivamente al gestore SM1 ed al gestore SM2 - l'ordine di eseguire un'operazione di push o un'operazione di pop.

In termini essenziali, il ramo superiore dell'architettura rappresentata nella figura 3, comprendente il gestore di stack SM1, è circa assimilabile - per quanto riguarda la struttura generale - alla struttura nota rappresentata nella figura 2. Per questo motivo, nella parte superiore della figura 3, per indicare segnali o parti identici o affini a quelli già descritti in precedenza con riferimento alla tecnica nota, sono stati utilizzati gli stessi riferimenti già riprodotti nella figura 2. E' pertanto superfluo ripetere in questa sede la relativa descrizione già fornita in precedenza.

Come elemento di differenza si apprezzerà però che, nello schema della figura 3, verso il modulo 12 è inviato, in unione al segnale data_bus, unicamente il segnale PC proveniente dal Program Counter e non già il segnale CCR che rappresenta il Condition Code Register.

Inoltre, nello schema della figura 3, il segnale ssp che rappresenta il puntatore allo stack di sistema è di fatto sdoppiato in due componenti ssp1 e ssp2 generate rispettivamente dal gestore di stack SM1 e dal gestore di stack SM2.

Il funzionamento dello schema rappresentato nella figura 3 prevede che i due gestori di stack SM1 e SM2, pilotati dall'unità di controllo UC, lavorino parallelamente per gestire, rispettivamente:

- l'uno (SM1), un primo stack ST1 all'interno di una memoria, quale una memoria RAM, utilizzata in modo convenzionale, e

- l'altro (SM2), un secondo stack, indicato con ST2, costituito da un banco di elementi di memoria (essenzialmente flip flop) appositamente creati ed in numero pari al numero di bit del segnale CCR che rappresenta il Condition Code Register per il numero degli interrupt del microcontrollore.

I gestori SM1 e SM2 operano generando due segnali stack pointer (appunto i segnali ssp1 e ssp2) il cui valore è sempre pari alla prima locazione di memoria utile all'interno del rispettivo stack ST1 o ST2.

In particolare, nella descrizione che segue, si assume che ad ogni operazione di push sugli stack corrisponda un decremento dello stack pointer associato. In modo duale, ad ogni operazione di pop corrisponde un incremento dello stack pointer associato.

L'unità di controllo UC genera quindi tre segnali per pilotare i due moduli gestori SM1 e SM2:

- un primo segnale up_down il cui scopo è discriminare le operazioni di push e pop,

- un segnale use_ss1 il cui scopo è abilitare esplicitamente l'operazione di push o pop sul primo stack ST1, e

- un segnale use_ss2 il cui scopo è abilitare esplicitamente l'operazione di push o pop sul secondo stack ST2.

Al verificarsi di un interrupt, l'unità di controllo UC pone il segnale up_down in modo da ottenere un'operazione di push. Contemporaneamente, pilota i segnali di scrittura dei due stack e pone i segnali use_ss1 e use_ss2 in modo da indicare ai

gestori di decrementare i propri stack pointer così da puntare la prossima localizzazione utile.

In modo duale, per il ripristino dello stato, al termine della routine di interrupt, l'unità di controllo UC pone il segnale up_down in modo da ottenere un'operazione pop. Contemporaneamente pilota i segnali use_ss1 e use_ss2 in modo da indicare ai gestori di incrementare i propri stack pointer così da puntare la locazione che contiene lo stato memorizzato da recuperare. Infine impone i segnali per la lettura dei due stack.

Per quanto riguarda le connessioni dei due stack in questione, si apprezzerà che le connessioni dello stack ST1 sono praticamente identiche a quello dell'unico stack ST rappresentato nella figura 2, inerente alla tecnica nota.

Lo stack ST2, cui è demandata la memorizzazione del segnale CCR, riceve invece in ingresso, oltre al segnale ssp2 già descritto in precedenza, un segnale data_stack_ccr appunto indicativo del valore del Condition Code Register, nonché due segnali rispettivamente di abilitazione (enable) e di latch la cui funzione è di abilitazione rispettivamente alla lettura ed alla scrittura dello stack ST2.

I riferimenti 14a e 14b indicano le corrispondenti linee di uscita degli stack ST1 e ST2 in cui sono presenti i relativi segnali di uscita in lettura.

Adottando l'architettura tradizionale rappresentata nella figura 2, tanto la fase di scrittura sullo stack di sistema quanto la fase di lettura dallo stack di sistema richiedono due periodi di clock, rispettivamente dedicati al Program Counter ed al segnale CCR. Ricorrendo invece all'architettura parallela rappresentata nella figura 3 tanto la fase di scrittura sullo stack quanto la fase di lettura dallo

stack può avvenire in modo parallelo per il Program Counter e per il CCR, occupando solo un periodo di clock.

Naturalmente, fermo restando il principio dell'invenzione, i particolari di realizzazione e le forme di attuazione potranno essere ampiamente variati rispetto a quanto descritto ed illustrato, senza per questo uscire dall'ambito della presente invenzione, così come definita dalle rivendicazioni annesse.

RIVENDICAZIONI

1. Procedimento per gestire in fase di interrupt uno stack di memoria associato ad un microcontrollore in funzione di un segnale di Program Counter (PC) e di un segnale di Condition Code Register (CCR) suscettibili di essere contenuti in rispettivi registri (PC, CCR), caratterizzato dal fatto che comprende le operazioni di:

- provvedere una prima parte di stack di memoria (ST1), comprendente un registro (PC) per detto segnale di Program Counter, ed una seconda parte di stack di memoria (ST2), costituita da un banco di elementi di memoria in numero pari al numero di bit di detto segnale di Condition Code Register (CCR) per il numero degli interrupt del microcontrollore, e

- far funzionare dette due parti di stack (ST1, ST2) in parallelo fra loro tramite rispettivi segnali di stack pointer.

2. Procedimento secondo la rivendicazione 1, caratterizzato dal fatto che comprende l'operazione di generare detti rispettivi segnali di stack pointer (ssp1, ssp2) con un valore pari alla prima locazione di memoria utile all'interno della rispettiva parte di stack (ST1, ST2).

3. Procedimento secondo la rivendicazione 1 o la rivendicazione 2, caratterizzato dal fatto che comprende, per pilotare dette due parti di stack (ST1, ST2), l'operazione di generare:

- un primo segnale (up_down) per discriminare fra operazioni di push e di pop,

- un secondo segnale (use_ss1), per abilitare esplicitamente l'operazione di push o pop su detta prima parte di stack di memoria (ST1), e

- un terzo segnale (use_ss2) per abilitare esplicitamente l'operazione di push o pop su detta seconda parte di stack di memoria (ST2).

4. Procedimento secondo la rivendicazione 3, caratterizzato dal fatto che, al verificarsi di un interrupt, comprende le operazioni di:

- porre detto primo segnale in modo da ottenere un'operazione di push,

- pilotare i segnali di scrittura di dette due parti di stack di memoria (ST1, ST2) ponendo detto secondo e detto terzo segnale in modo da decrementare i rispettivi stack pointer così da puntare la successiva locazione utile.

5. Procedimento secondo la rivendicazione 3 o la rivendicazione 4, caratterizzato dal fatto che, per il ripristino dello stato al termine di un interrupt, comprende le operazioni di:

- porre detto primo segnale in modo da ottenere un'operazione di pop,

- pilotare detto secondo e detto terzo segnale in modo da incrementare gli stack pointer di detta prima e detta seconda parte di stack (ST1, ST2) così da puntare la locazione contenente lo stato memorizzato da recuperare, e

- imporre i segnali per la lettura di dette due parti di stack (ST1, ST2).

6. Dispositivo per gestire in fase di interrupt uno stack di memoria associato ad un microcontrollore in funzione di segnale di Program Counter (PC) e di un segnale di Condition Code Register (CCR) suscettibili di essere contenuti in rispettivi registri (PC, CCR), caratterizzato dal fatto che comprende:

- una prima parte di stack di memoria (ST1), comprendente un registro (PC) per detto segnale di Program Counter, e

- una seconda parte di stack di memoria (ST2), costituita da un banco di elementi di memoria in numero pari al numero di bit di detto segnale di Condition Code Register (CCR) per il numero degli interrupt del microcontrollore, e

- almeno un modulo di gestione (UC, SM1, SM2) configurato per far funzionare dette due parti di stack (ST1, ST2) in parallelo fra loro tramite rispettivi segnali di stack pointer (ssp1, ssp2).

7. Dispositivo secondo la rivendicazione 6, caratterizzato dal fatto che comprende almeno un modulo di gestione (UC, SM1, SM2) configurato per generare detti rispettivi segnali di stack pointer (ssp1, ssp2) con un valore pari alla prima locazione di memoria utile all'interno della rispettiva parte di stack (ST1, ST2).

8. Dispositivo secondo la rivendicazione 6 o la rivendicazione 7, caratterizzato dal fatto che comprende almeno un modulo di gestione (UC, SM1, SM2), per pilotare dette due parti di stack (ST1, ST2), detto almeno un modulo di gestione (UC, SM1, SM2) essendo configurato per generare:

- un primo segnale (up_down) per discriminare fra operazioni di push e di pop,

- un secondo segnale (use_ss1), per abilitare esplicitamente l'operazione di push o pop su detta prima parte di stack di memoria (ST1), e

- un terzo segnale (use_ss2) per abilitare l'operazione di push o pop su detta seconda parte di stack di memoria (ST2).

9. Dispositivo secondo la rivendicazione 8, caratterizzato dal fatto che, al verificarsi di un interrupt, detto almeno un modulo di gestione (UCC, SM1, SM2) è configurato per:

- porre detto primo segnale, in modo da ottenere un'operazione di push,

- pilotare i segnali di scrittura di dette due parti di stack di memoria (ST1, ST2) ponendo detto secondo e detto terzo segnale in modo da decrementare i rispettivi stack pointer così da puntare la successiva locazione utile.

10. Dispositivo secondo la rivendicazione 8 o la rivendicazione 9, caratterizzato dal fatto che, per il ripristino dello stato al termine di un interrupt, detto almeno un modulo di gestione (UCC, SM1, SM2) è configurato per:

- porre detto primo segnale in modo da ottenere un'operazione di pop,

- pilotare detto secondo e detto terzo segnale in modo da incrementare gli stack pointer di detta prima e detta seconda parte di stack (ST1, ST2) così da puntare la locazione contenente lo stato memorizzato da recuperare, e

- imporre i segnali per la lettura di dette due parti di stack (ST1, ST2).

11. Dispositivo secondo una qualsiasi delle precedenti rivendicazioni 6 a 10, caratterizzato dal fatto che comprende:

- un'unità di controllo (UC) comune a dette due parti di stack (ST1, ST2),

- due moduli di gestione (SM1, SM2) rispettivamente associati ciascuno ad una di dette due parti di stack (ST1, ST2) per inviare verso dette due parti di stack (ST1, ST2) detti rispettivi segnali di stack pointer (ssp1, ssp2) in funzione di rispettivi segnali di esecuzione di operazione (use_ss1, use_ss2) generati da detta unità di controllo comune (UC).

12. Dispositivo secondo la rivendicazione 8 e la rivendicazione 11, caratterizzato dal fatto che detta

unità di controllo (UC) è configurata per inviare detto primo segnale (up_down) verso detti due moduli di gestione (SM1, SM2).

13. Dispositivo secondo una qualsiasi delle rivendicazioni 6 a 12, caratterizzato dal fatto che detti elementi di memoria in detta seconda parte di stack di memoria (SM2) sono costituiti da flip flop.

14. Prodotto informatico direttamente caricabile nella memoria di un elaboratore numerico e comprendente porzioni di codice software suscettibili di attuare il procedimento secondo una qualsiasi delle rivendicazioni 1 a 5 quando detto prodotto è eseguito su un elaboratore numerico.

RIASSUNTO

Per gestire in fase di interrupt uno stack di memoria associato ad un microcontrollore in funzione di un segnale di Program Counter (PC) e di un segnale di Condition Code Register (CCR) suscettibili di essere contenuti in rispettivi registri (PC. CCR), si provvede una prima parte di stack di memoria (ST1) comprendente un registro (PC) per il segnale di Program Counter, ed una seconda parte di stack di memoria (ST2) costituita da un banco di elementi di memoria in numero pari al numero di bit del segnale di Condition Code Register (CCR) per il numero degli interrupt del microcontrollore. Le due parti di stack (ST1, ST2) sono fatte funzionare in parallelo fra loro tramite rispettivi segnali di stack pointer (ssp1, ssp2).

(Figura 3)

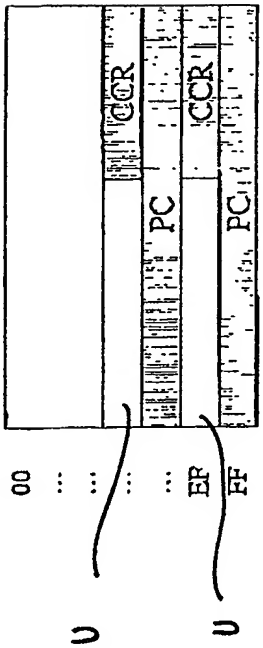
00
EF
FF
U
U
CCR
PC
CCR
PC

Fig.1

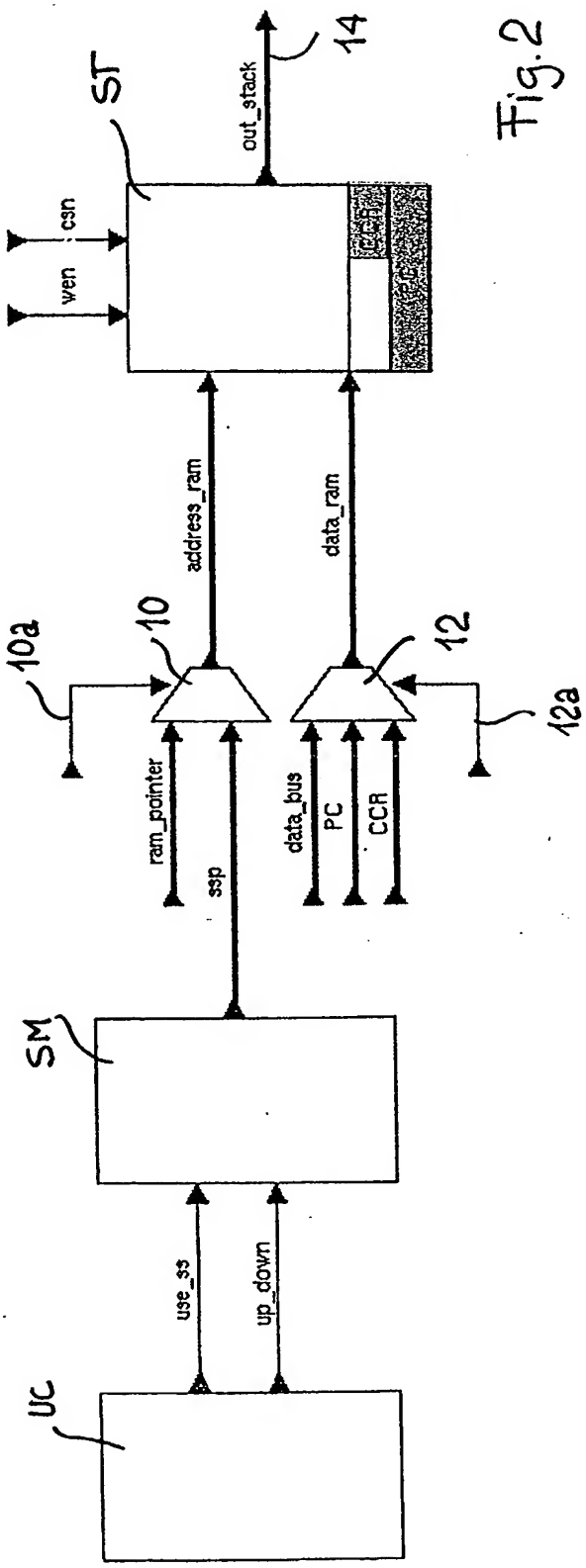
UC
SM
ST
use_ss
up_down
ssp
ram_pointer
data_bus
PC
CCR
address_ram
data_ram
csn
wen
out_stack
10
10a
12
12a
14

Fig.2

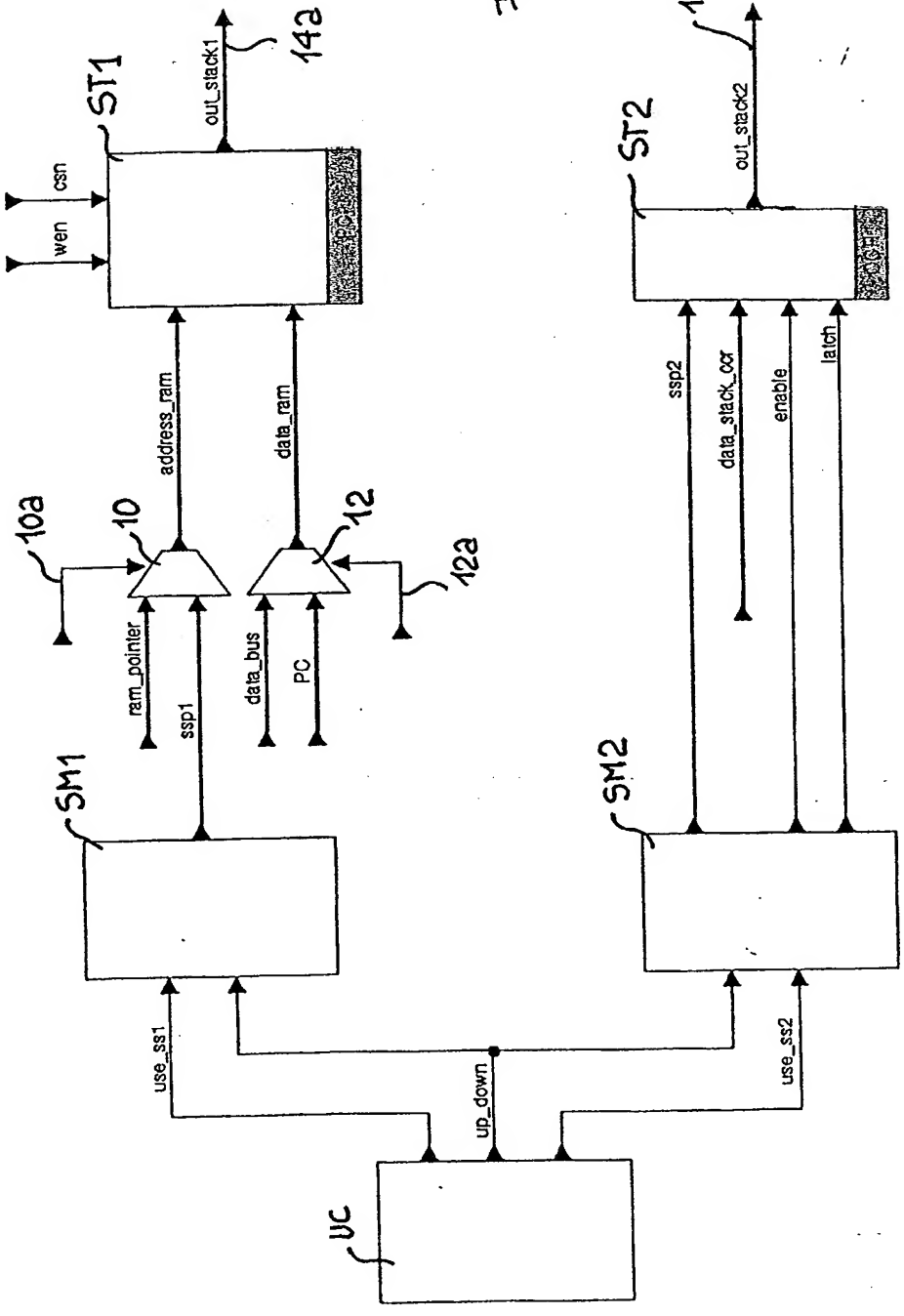
ST1
out_stack1
14a
csn
wen
address_ram
data_ram
10a
10
12
12a
ram_pointer
ssp1
data_bus
PC
SM1
ST2
out_stack2
14b
ssp2
data_stack_cor
enable
latch
SM2
use_ss1
up_down
use_ss2
UC

Fig. 3